# Design Overview for Solitaire

Name: Minh Nguyen

Student ID: 104309099

# Summary of Program

Describe what you want the program to do… one or two paragraphs.

Include a sketch of sample output to illustrate your idea.

# The Solitaire game is a classic single-player card game where the player's objective is to move all the cards to a foundation area, following specific rules for card movement. The game will have a graphical user interface (GUI) that allows the player to interact with the cards on the screen. The game will start with a shuffled deck of cards dealt onto the tableau in seven piles. The player will then make moves by dragging and dropping cards according to the rules of Solitaire.

**Sample Output Sketch:**

-----------------------

Solitaire - Minh Nguyen

-----------------------

Foundation Piles: [empty] [empty] [empty] [empty]

--------------------------------------------------

Stock: [remaining cards] Waste: [discarded cards]

Tableau:

Pile 1: [face-down cards] [face-up cards]

Pile 2: [face-down cards] [face-up cards]

...

Pile 7: [face-down cards] [face-up cards]

Drag and drop cards to make moves.

Press 'R' to restart the game.

# Required Data Types

Describe each of the records and enumerations you will create using the following table (one per record).

Table 1: Card details

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Notes |
| suit | String | Represents the suit of the card (e.g., "Clubs," "Hearts"). |
| value | String | Represents the value of the card (e.g., "A" for Ace, "2" for Two). |
| is\_face\_up | Boolean | Indicates whether the card is face-up (True) or face-down (False). |
| Image\_file\_name | String | File path to the card's image when it’s face up. |

Table 2: Pile Details (Enumeration)

|  |  |
| --- | --- |
| Value | Notes |
| FOUNDATION | Represents the foundation piles where cards are built up. |
| TABLEAU | Represents the tableau piles where cards can be moved. |
| STOCK | Represents the stock pile where remaining cards are drawn from. |
| WASTE | Represents the waste pile where discarded cards are placed. |

Table 3: Screen Dimensions

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Notes |
| SCREEN\_WIDTH | Integer | Width of the game screen (e.g., 1024 pixels). |
| SCREEN\_HEIGHT | Integer | Height of the game screen (e.g., 768 pixels). |
| SCREEN\_TITLE | String | Title of the game window ("Drag and Drop Cards"). |

Table 4: Card Sizing Constants

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Notes |
| CARD\_SCALE | Float | Scale factor for card size (e.g., 0.6 for 60% scale). |
| CARD\_WIDTH | Float | Width of a card (scaled). |
| CARD\_HEIGHT | Float | Height of a card (scaled). |

Table 5: Mat Sizing Constants

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Notes |
| MAT\_PERCENT\_OVERSIZE | Float | Oversize factor for mat size (e.g., 1.25 for 125% size). |
| MAT\_HEIGHT | Integer | Height of the mat (calculated based on CARD\_HEIGHT and MAT\_PERCENT\_OVERSIZE). |
| MAT\_WIDTH | Integer | Width of the mat (calculated based on CARD\_WIDTH and MAT\_PERCENT\_OVERSIZE). |

Table 6: Margin Constants

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Notes |
| VERTICAL\_MARGIN\_PERCENT | Float | Vertical margin as a percentage of MAT\_HEIGHT (e.g., 0.10 for 10%). |
| HORIZONTAL\_MARGIN\_PERCENT | Float | Horizontal margin as a percentage of MAT\_WIDTH (e.g., 0.10 for 10%). |

Table 7: Pile Positions and Spacing Constants

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Notes |
| BOTTOM\_Y | Float | Y-coordinate of the bottom row of piles. |
| START\_X | Float | X-coordinate of where to start placing piles on the left side. |
| TOP\_Y | Float | Y-coordinate of the top row of piles. |
| MIDDLE\_Y | Float | Y-coordinate of the middle row of piles. |
| X\_SPACING | Float | Horizontal spacing between each pile. |

Table 8: Card Values and Suits Constants

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Notes |
| CARD\_VALUES | List of Strings | List of card values (e.g., ["A", "2", ..., "K"]). |
| CARD\_SUITS | List of Strings | List of card suits (e.g., ["Clubs", "Hearts", ...]). |

Table 9: Card Vertical Offset Constant

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Notes |
| CARD\_VERTICAL\_OFFSET | Float | Vertical offset for fanning stacked cards (scaled). |

Table 10: Face Down Card Image

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Notes |
| FACE\_DOWN\_IMAGE | String | File path to the image of the face-down card. |

Table 11: Pile Constants (Enumeration)

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Notes |
| PILE\_COUNT | Integer | Total number of piles in the game (e.g., 13 piles). |
| BOTTOM\_FACE\_DOWN\_PILE | Integer | Index of the bottom face-down pile. |
| BOTTOM\_FACE\_UP\_PILE | Integer | Index of the bottom face-up pile. |
| PLAY\_PILE\_1 | Integer | Index of the first play pile. |
| PLAY\_PILE\_2 | Integer | Index of the second play pile. |
| PLAY\_PILE\_3 | Integer | Index of the third play pile. |
| PLAY\_PILE\_4 | Integer | Index of the fourth play pile. |
| PLAY\_PILE\_5 | Integer | Index of the fifth play pile. |
| PLAY\_PILE\_6 | Integer | Index of the sixth play pile. |
| PLAY\_PILE\_7 | Integer | Index of the seventh play pile. |
| TOP\_PILE\_1 | Integer | Index of the first top pile. |
| TOP\_PILE\_2 | Integer | Index of the second top pile. |
| TOP\_PILE\_3 | Integer | Index of the third top pile. |
| TOP\_PILE\_4 | Integer | Index of the fourth top pile. |

# Overview of Program Structure

List the main functions/procedures you are going to need to create this program. For each function/procedure provide its name and a brief description of what it will do.

Don’t spend too long on this at this stage. Focus on the main things you think you are likely to need and you can build on this as your program develops.

Include a structure chart (once you have your proposal approved by your tutor)

**Main Functions/Procedures**

1. setup(): Initializes the game by creating card sprites, shuffling the deck, and setting up piles.
2. on\_draw(): Renders the game screen, including mats, cards, and information.
3. on\_key\_press(symbol, modifiers): Handles key presses, such as restarting the game.
4. on\_mouse\_press(x, y, button, key\_modifiers): Handles mouse clicks on cards, mats, and piles.
5. on\_mouse\_release(x, y, button, modifiers): Handles the release of the mouse button after dragging cards.
6. on\_mouse\_motion(x, y, dx, dy): Updates the positions of dragged cards while the mouse is in motion.
7. pull\_to\_top(card): Moves a card to the top of the rendering order.
8. remove\_card\_from\_pile(card): Removes a card from its current pile.
9. get\_pile\_for\_card(card): Determines which pile a card belongs to.
10. move\_card\_to\_new\_pile(card, pile\_index): Moves a card to a new pile based on the pile index.

**Program Flow**

* The program initializes by creating card sprites, mats, and piles.
* The game is rendered on the screen with all the necessary elements.
* The player can interact with the cards by clicking and dragging them.
* Cards can be moved between tableau piles, foundation piles, and waste piles according to Solitaire rules.
* The game checks for win conditions, and if met, displays a victory message.
* The player can restart the game by pressing the 'R' key.

Ảnh có chứa ảnh chụp màn hình, hàng, hình vuông, thiết kế

Mô tả được tạo tự động